Preserving Context in a Multi-tasking Clinical Environment: A Pilot Implementation

Dean F. Sittig, Ph.D., Jonathan M. Teich, M.D., Ph.D.,
Joel A. Yungton, B.S., Henry C. Chueh, M.D.

Clinical Systems Research and Development, Partners Healthcare System, Boston, MA

ABSTRACT

The Partners Clinical Application Suite (CAS) is a multi-tasking software architecture that facilitates the development, deployment, and use of advanced clinical information management applications. This paper describes 1) a software shell in which clinical applications run; 2) an application programming interface (API); and 3) development of a set of "Look & Feel" guidelines. Through its emphasis on support for multi-tasking and application interoperability, CAS facilitates preservation of the user's context.

INTRODUCTION

Over the past 30 years various investigators have struggled with the problem of developing easy to use, yet full-function, clinical computing environments. In their landmark 1982 report, Matheson and Cooper[1] found that academic health centers had information systems consisting of "fragmented mixtures of single function, manual, and computer-based files that [could] neither communicate [nor] exchange information effectively."

Medical providers, on the other hand, often do several tasks at once, in the care of a patient[2]. A provider may need to look up a patient's lab results, consider them in the setting of the patient's past history and problem list, and write orders based on the combined information[3]. Similarly, software developers in general have shifted to suites of applications which work as a unit to handle the diverse information needs that occur in the clinical workday[4].

Over the coming years, the concepts of pragmatic integration and system agility will be keys to the successful design, development and implementation of advanced clinical information systems. Pragmatic integration implies that few organizations can afford to stop and rewrite all of their applications to conform to a new software model. Therefore, developers must come up with new software architectures that allow disparate applications to run within an integrated framework.

System agility is the ability of a system to evolve rapidly to meet changing requirements. Ideally, the system should be composed of blocks which are as modular and separate as possible, permitting rapid change to individual components, yet continue to function in an integrated fashion.

Early on we determined that relying solely on the WIN-95 operating system as our application shell was not satisfactory for the following reasons: 1) the flexibility of the interface makes training for occasional users difficult; 2) maintaining a common look and feel across applications for the patient identifying information display is difficult; 3) providing a constant visual display of all available applications is impossible.

Therefore, we are developing a new clinical computing architecture that addresses each of these issues. This paper describes the Partners Clinical Application Suite (CAS), an architecture that facilitates the development, deployment, and use of advanced clinical information management applications. Through its emphasis on support for multi-tasking and application interoperability, CAS facilitates preservation of the user's context.

BACKGROUND

Partners Healthcare System

Partners Healthcare currently consists of the Massachusetts General and Brigham & Women's hospitals, Dana-Farber/Partners Cancer Care, and North Shore Medical Center. Partners' practices care for over 800,000 patients in the greater Boston area.

The Partners Computing Environment

The Partners computing environment is based on a three-tier client-server architecture. The data and application services tiers are based on MUMPS running on Windows NT on multi-processor Pentium

1091—8280/97/$5.00

Pro servers. These layers provide services to the user interface layer, which primarily uses Visual Basic, but also supports native MUMPS and Web-browser interfaces. It is projected that by early 1998 there will be 15 machines at the database layer serving over 100 gigabytes of data, 30 machines at the application services layer, and over 20,000 client machines.

PROJECT GOALS

The clinical workstation project's goals can be divided into two distinct categories. The first concerns the software environment with which the clinical users will interact. Users must be able to multi-task without getting lost. The second concerns the ability of the IS staff to develop, maintain, and control the various clinical applications.

Issues in a Multi-tasking Clinical Environment

1) Applications must work together as interoperable tools. Clinicians must be able to switch among several clinical applications without losing track of where they are or having to back out of, or shutdown, the application they are leaving.

2) The system must ensure that all concurrent applications are focused on the same patient. In a multi-tasking environment with multiple patients, it is easy for the user to become confused and enter data for the wrong patient.

3) All applications should maintain a common look and feel. All applications must have common metaphors, button locations and actions, screen navigation, and help functions.

4) Users must be able to run third-party applications (Scientific American Medicine, Netscape browser, etc.), concurrently with, and (where possible) interactively with locally-developed applications.

5) There should be a visual 'anchor' to help users maintain their frame of reference.

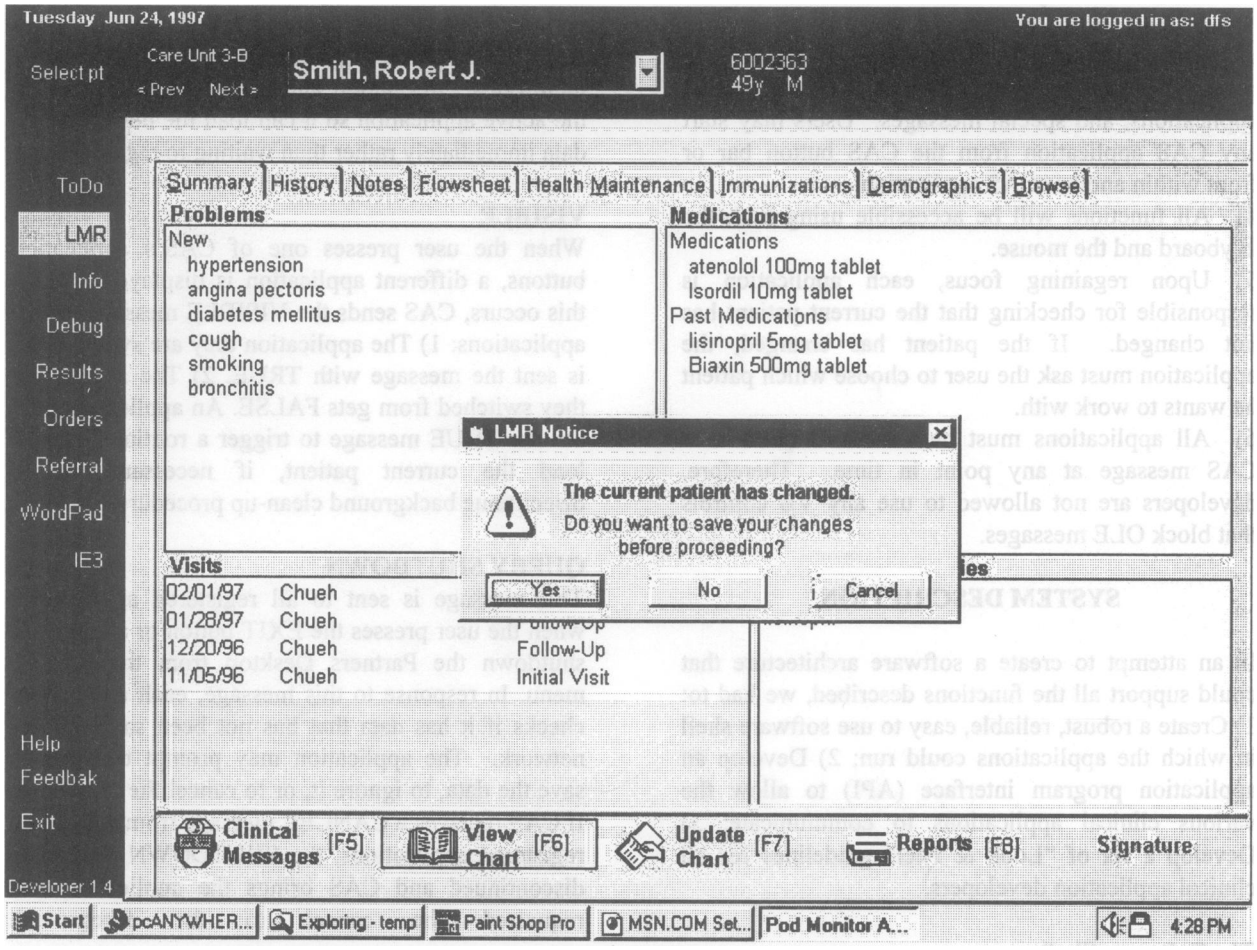

Figure 1. The Clinical Application Suite. The user has just changed back to the Longitudinal Medical Record (LMR) from Orders. CAS prompts the user to save pending data entries on a previous patient.

Issues in Software Development

1) Applications must be able to be developed, deployed, and updated separately, while the overall system keeps running.
2) The system should facilitate the sending and receiving of manual or automated alerts and messages related to clinical events and patients.
3) Users should not have to re-enter their password before starting each individual application.
4) The system should facilitate the saving of uncompleted work in the event of either a change in current patient, system time-out, or system shutdown.

In an attempt to insure that all the project goals could be met, the following design decisions were made.

Design Decisions for the Clinical Application Suite

1) To ensure that users are not missing some of the information presented by an application, all applications will run in a constant-size window. CAS changes the desktop workspace to ensure this.
2) Only one copy of each application will be allowed to run at any one time.
3) A small portion of the screen will be reserved for the CAS itself: this includes space to display the patient identifier, the available and active applications, and special messages. Users may start any CAS application from the CAS button bar or from within another CAS application.
4) All functions will be accessible using both the keyboard and the mouse.
5) Upon regaining focus, each application is responsible for checking that the current patient has not changed. If the patient has changed, the application must ask the user to choose which patient he wants to work with.
6) All applications must be able to respond to a CAS message at any point in time. Therefore, developers are not allowed to use any VB controls that block OLE messages.

SYSTEM DESCRIPTION

In an attempt to create a software architecture that could support all the functions described, we had to: 1) Create a robust, reliable, easy to use software shell in which the applications could run; 2) Develop an application program interface (API) to allow the various clinical applications to communicate; 3) Develop a set of "Look & Feel" guidelines for all clinical application developers.

The Software Shell

The current version of the CAS utilizes many features of the WIN-95 desktop and operates closely with the WIN-95 operating system. The CAS executable is implemented using an application desktop toolbar (appbar) in the same manner as the Microsoft Office shortcut bar[5]. Once CAS startes, it changes the size of the desktop reported to all other programs (via a Windows system call) and anchors itself to the top and left side of the desktop. By setting the appbar to be "always on top" any other WIN-95-based program can run and appear visually to be "in the CAS". Each application that runs within CAS is a stand-alone *.exe with an exposed OLE class which CAS instantiates and uses to send messages. Each individual application that is started must register with CAS. Once registered, it can receive messages from the CAS.

The Application Programming Interface Messages

The following messages were created to allow applications to communicate with CAS.

PATIENT CHANGED

When the user or an application changes the current patient, the PATIENT CHANGED message is sent to CAS. CAS responds by displaying the new patient's identifying information and sending the message to the active application so it can load the new patient's data immediately rather than waiting to regain focus.

VISIBLE

When the user presses one of CAS's application buttons, a different application is displayed. When this occurs, CAS sends the VISIBLE message to two applications: 1) The application they are switching to is sent the message with TRUE. 2) The application they switched from gets FALSE. An application may use the TRUE message to trigger a routine that will load the current patient, if necessary, or to discontinue background clean-up procedures.

QUERY SHUTDOWN

This message is sent to all registered applications when the user presses the EXIT button or attempts to shutdown the Partners Desktop from the START menu. In response to this message, each application checks if it has data that has not been saved to the network. The application may prompt the user to save the data, to ignore it, or to cancel the shutdown. If CAS receives a CANCEL message from any of the registered applications, the SHUTDOWN process is discontinued and CAS brings the application that requested the cancel to the foreground.

Table 1. Overview of CAS API and Application responsibilities.

Initiated by Desktop	CAS Response	Application Action
System shutdown	send SHUTDOWN	save data; exit
Time out	send SCREEN SAVE ON	save data
Initiated by User		
Change application	send VISIBLE=TRUE	verify current patient
Select Patient	Display new patient send PATIENT CHANGED	Get new patient data
Shutdown	send QUERY SHUTDOWN	Inform user; save data; exit
Initiated by Application		
Change application	receive ACTIVATE APP; change to new app	verify current patient
NOTIFY ALL	relay message to all apps	take appropriate action
REGISTER with CAS	instantiate app messaging class	perform startup routine
New data available	send NEWDATA via NOTIFY ALL	Get new patient data

SHUTDOWN
Sent to all applications when CAS receives a message from the Partners Desktop that the workstation has timed out and it is being reset for the next user. Each application should treat the message in much the same manner as a Query ShutDown message except that the user is not available to respond to queries. Each application must decide on its own whether to save any partial data or discard it. Whether the application saves data or not, it must unload itself and allow CAS to continue the normal shutdown process.

SCREEN SAVE ON / SCREEN SAVE OFF
The screen saver is activated after a time-out period; the system does not shut down, but it must be reactivated by the current user to continue work. Another user may make use of the workstation to start a new session. The applications should take advantage of this state to save any data in anticipation of a user change. An application may also choose to start/end any background cleanup or data retrieval processes at this time.

REGISTER
Applications that are written in-house are required to register with CAS upon loading. CAS uses the application's messaging class to send messages, and the application's window handle to position it within CAS's boundaries.

ACTIVATE APP
Applications send this message to change the current application. The effect is the same as if the user had selected another application button.

NOTIFY ALL
Applications send this message to ask CAS to broadcast a message to all registered applications. For example, when an application that is registered with CAS receives new data relevant to other clinical applications, the application uses NOTIFY ALL to send the NEWDATA message to CAS. The application may decide to: a) reload the current patient's data, b) query the user as to whether or not they wish to reload the current patient's data, or c) ignore the message altogether.

Development of the Look and Feel guidelines
A committee was named to develop overall Look and Feel guidelines (L&F) . The committee was composed of clinical end users, experienced clinical software developers, technical experts, and several informaticians. Early meetings emphasized demonstration and discussion of what was working and not working in the current clinical system. Later meetings focused on the capabilities of the new graphical user interface development tools that had been chosen as the standard software development environment. Once development of the various clinical applications began, members of the L&F committee were assigned to each of the projects to work with each project team[6]. As new problems and solutions were identified, these were documented and distributed via an internal WEB site.

DISCUSSION

Over the past year, we have made significant progress in developing the software architecture for the new Partners clinical applications. Along the way we have learned many valuable lessons as the result of our experimentation with various software architectures.

In the first working prototype of the Partners Clinical Application Suite (CAS) each of the clinical applications was written as an in-process dynamic link library (*.DLL). In an attempt to make the CAS application (including the in-process DLLs) look and feel like one big application, we used a low-level WIN-95 call to "switch the parent" of the window containing the graphical elements of the clinical application (i.e., the DLL) to a window on a TAB control that was used within the CAS to implement task switching. This WIN-95 feature allowed us to control the position of each clinical application and ensure that all applications were tightly integrated.

An early version of the CAS used the same "switch the parent" call although each of the clinical applications were written as out-of-process DLLs. While this implementation worked, the time required to load the individual clinical applications was 2-3 times longer (3-4 secs) than the in-process DLL model.

We were forced to abandon both of these early implementations for the following reasons: 1) Both keyboard and mouse actions were unpredictable as the user moved among the various applications; 2) Neither model allowed developers to implement application modal windows within applications; 3) Both models placed an inordinate burden on application developers to learn about and conform to the CAS-concept of creating programs; 4) When an individual application crashed, recovery was difficult. 5) This model made it more difficult for naïve users to run third-party applications while still keeping track of their clinical work.

The Clinical Context Object Working Group[7] (CCOW) is developing a specification for a patient context object that has the potential to allow clinical applications created by different vendors to work in much the same manner as the CAS. They are currently struggling with several clinical scenarios that are not current concerns in our development effort: 1) managing the current patient if the user starts two versions of the same application, and 2) handling the possibility that applications will be using two or more different patient databases and that the current patient many not be registered in one of the databases.

CONCLUSION

A large, integrated clinical information system is composed of many separate application programs. Each performs a small set of highly specialized clinical functions on the patient database. Developing a multi-tasking environment in which each of these applications can operate is difficult. The key requirements for such an environment are:

1) maintain the patient context across applications;
2) maintain a common look and feel across applications;
3) maintain the user identification and authorization status across applications;
4) facilitate the development and deployment of new applications to clinical users.

We implemented a multi-faceted approach including development of a common patient object, a common user object, a detailed set of look and feel guidelines, and a custom software architecture to satisfy these requirements.

References

1. Matheson NW, Cooper JAD. Academic information in the health sciences center: roles for the library in information management. J Med Educ 57:1-93; 1982.
2. Kuhn K, Reichert M, Nathe M. et al. An infrastructure for cooperation and communication in an advanced clinical information system. Proc Symp Comput Appl Med Care (1994):519-23.
3. Tang PC Annevelink J Fafchamps D et al. Physicians' workstations: integrated information management for clinicians. Proc Symp Comput Appl Med Care (1991) :569-73.
4. Chueh HC, Raila WF, Pappas JJ et al. A component-based, distributed object services architecture for a clinical workstation. Proc Symp Comput Appl Med Care (1996):638-42.
5. Richter J. Extend the Windows 95 Shell with Application Desktop Toolbars. Microsoft Systems Journal Vol. 3 (1996).
6. Hopper S, Hambrose H, Kanevsky P. Real world design in the corporate environment: Designing an interface for the technically challenged. Proc. Comp. & Human Interaction (1996):489-495.
7. Clinical Context Object Working Group www.mcis.duke.edu:80/standards/clin-cntxt